Experiments in Digital Video for Workstations

Ian Wilson, David Milway and Andy Hopper
Olivetti Research Laboratory
24a, Trumpington Street
Cambridge CB2 1QA
England

March 1989

Abstract

This paper describes experimental work in the use of digital video by computer workstations. The approach is essentially practical, since the aims of the research were to gain experience with the handling of video, and to investigate some of the systems aspects of its integration into the workstation environment. The motivation for the work is explained, along with the compromises which were necessary to make the work possible. The hardware and software tools used in the experimentation are described, and the experiences gained from the first trial applications are discussed.

1. Introduction

Since its foundation in 1986, the Cambridge Olivetti Research Laboratory has been working heavily in the area of digital video. The work is based on the use of a high speed local area network with sufficient capacity to carry video traffic [Hopper88]. It also owes much to work done at the University of Cambridge Computer Laboratory, primarily the Cambridge Model Distributed System [Needham82] and the ISLAND multi-media project [Want88].

The main aim of our work was to investigate the use of video in a computer environment. As well as the problems presented by the design of hardware and software to handle the video data, there was the more fundamental question of whether computer video was worthwhile. Were video applications desirable? How could video be used to enhance current applications? Also, what kinds of new applications could be produced to take advantage of computer video? As part of our work, we hoped to find answers to some of these questions.

At the beginning of the work, two related decisions were taken which made our approach different to that going on elsewhere. Firstly, all video was to be handled digitally, with the video traffic being carried by the Local Area Network in addition to its normal load. This contrasts with the work done at MIT [Davenport88, Lampe88], where all video is carried along analogue lines and switched by analogue switches. Secondly, once digitized, we wanted to be able to store, edit and retrieve video information. This meant that local video display hardware, for example [Parallax87] which digitizes directly into a frame buffer, was inappropriate for our needs. Handling video digitally would mean that it could be stored in conventional filing systems and edited in the same way as text; and, although the memory, disc storage and network bandwidth requirements for video seem prohibitive now, future technological developments in these areas should remove the restrictions.

Section 2 of this paper deals with some of the fundamental decisions which were taken to enable us to perform the video work. Section 3 describes the hardware and software tools which were used in the experimentation, and section 4 describes the first three video applications which were built. Section 5 discusses our experiences while doing the video work, and the feedback received by the users of the applications. In conclusion, section 6 evaluates the work, and describes some of the areas which remain to be tackled.

2. Fundamental Decisions

The bandwidth required for the transmission of high quality digital images is large—200 Mbps is not an untypical figure—and our 75 Mbps LAN was not capable of carrying this kind of traffic. Similarly, when considering the retention of digital images in some kind of file store, the sheer amount of video data is such that conventional discs are both too small and too slow.

One way round the problem of handling large quantities of video data is to apply compression techniques, so as to reduce the problem to a more manageable size. Given the amount of compression work being undertaken by other people, particularly in the area of video CODECs for ISDN (for example, COST 211), it was decided that we should start developing our video systems immediately, but in such a way that we could take advantage of compression products when they became available. In other words, we wanted the advent of compression to enhance the work we had done, rather than be a prerequisite for the work to start.

It soon became obvious that certain compromises would be necessary if we were to start video work without the benefits of compression. Basically, the video data rate would have to be reduced by a sufficient amount so as to make the experimentation possible on the available equipment. Having said that, there are many ways of altering video information so as to reduce the required bandwidth, and not all are equally acceptable to the viewer. It was essential that the data rate was reduced, but without making the final effect off-putting to the potential users.

Without using compression techniques, there are three ways in which the data rate required by a video signal can be reduced. Firstly, the size of the image can be shrunk; so, rather than using full resolution TV quality images, the size can be reduced to 256×256 (or even 128×128). Secondly, the depth of the image can be reduced, for example by using monochrome rather than colour images. Finally, the frame update rate can be cut from TV quality (25 fps in Europe, 30 fps in the USA) to something much slower, say 5 fps, so as to allow more time for picture transmission.

As mentioned earlier, although each of the techniques allows the effective video bit rate to be reduced, they have very different effects on the user of the system. Making the image smaller is perfectly acceptable, since the *quality* of the image is maintained, even though it occupies a smaller area on the screen. The only disadvantage of a small picture is that, because the resolution of the image is reduced, there could be problems when dealing with very fine detail, for example text on a sheet of paper. Using monochrome rather than colour images may lose some information for the user, but in the vast majority of cases is perfectly acceptable. It seems that, apart from an initial disappointment (most people seem to expect colour) monochrome images are well suited for video experimentation. Reducing the frame rate, however, makes a very distinct impression on the user. Smooth motion becomes jerky, lip synchronisation is lost, and assuming that the images are being transmitted over a network, the delays involved tend to be large. In our opinion, this was not an acceptable way to reduce the video bandwidth, so in all the work described here, a minimum frame update rate of 25 fps is assumed.

3. Experimental Tools

In this section, we describe some of the hardware and software tools which enabled us to perform the early video experimentation.

3.1 Network

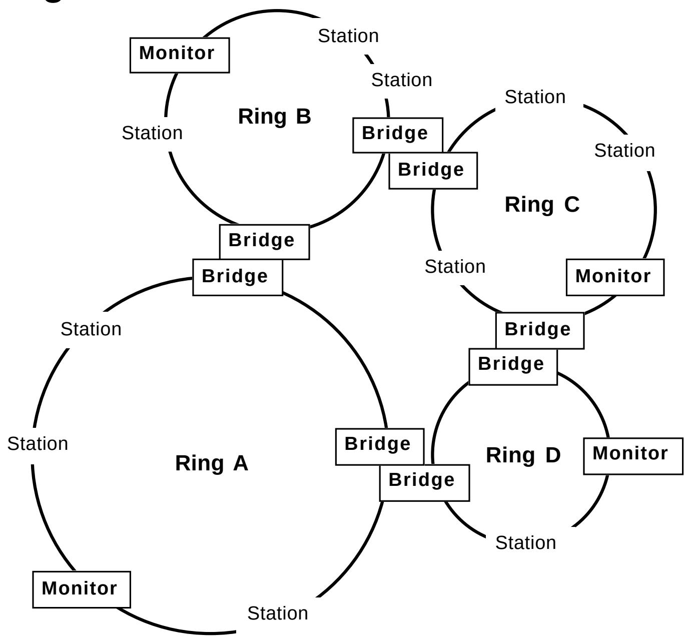
Following on from the slower 10 Mbps Cambridge Ring, a project was started at the University of Cambridge to produce a much enhanced version, the Cambridge Fast Ring (CFR), designed to work at 75 MHz and implemented entirely in VLSI.

The CFR is a slotted ring with 16 bit node addresses and 256 bit data units. The protocol ensures that each station is only allowed the use of one slot in any one ring revolution, and that after using a slot, it must free it and pass it on before attempting to transmit again. In this way, every station is guaranteed a certain proportion of the available bandwidth. Also, there is a defined worst-case delay between transmissions, related to the number of slots in the ring and the number of attached stations. Being a slotted ring, though, the maximum point-to-point bandwidth is fixed by the slot structure of the ring and its access protocol. A slotted ring, however, is well suited for experimental video work, since it gives a relatively high throughput, in addition to providing very good low level sharing of the available bandwidth. Also, being a packet network with small data units, the CFR is capable of dealing with the kinds of bursty traffic generated by many of the rate-adaptive video encoding methods [Coudreuse88].

The Cambridge Fast Ring is implemented as two VLSI chips. The *repeater* chip, built using ECL technology, handles the phase locking of incoming and outgoing signals, and performs the conversion of the CFR data from its serially encoded form on the ring, to its 8 bit parallel form used within the network node. The *station* chip, built using 2.4µm CMOS technology, performs all the functions expected of a network node: primarily data transmission and reception. Included on the station chip are two FIFOs, the size of a CFR packet, one each for reception and transmission. The station chip is also capable of performing two other important functions. By altering a configuration pin on the chip, the chip becomes a *monitor* (of which there must be one per ring) which controls the reframing of the ring, maintenance of the slot structure, and the reporting of error conditions. Another alteration of the configuration pin turns the station chip into a *bridge*; the connection of two such chips back-to-back allows for the transparent movement of packets from one ring to another.

Several different types of host interface for the CFR exist including: PC, VME, Archimedes and Transputer. All are relatively simple, in that they present a programmed interface to the host bus, with one interrupt per packet received or transmitted, and no DMA capability. For video experimentation, the only disadvantage of this simplicity is that a very fast processor is required to control the CFR node. It was this requirement that defined the decision as to which host computer to use for our video work.

Cambridge Fast Ring Topology



3.2 Host Computer

There were several stringent criteria for which host computer to use for the video experimentation. Most importantly, the processor had to be fast enough to keep up with the CFR interface, both in the sense of data throughput and interrupt latency. Bearing in mind that we would want to display the video picture once it had been received, a frame buffer capable of displaying reasonable quality monochrome images was essential. Also very important to the work was that it should be easy to build extension hardware for the computer.

Of the computers available on the market at time the research was started, the Acorn Archimedes [Acorn87] matched all the criteria, and had the advantage of being relatively cheap. The Archimedes is based on the Acorn Risc Machine (ARM) processor, delivering 3 to 4 MIPS. As well as being fast, the processor has a very low interrupt latency, and variety of screen formats, including greyscale. Although supporting 8 bit pixels, this cannot not be used to generate 256 grey levels—the best monochrome image which can be generated is only 4 bits deep. Also the ARM does not support Direct Memory Access, so all data transfers to and from peripherals must be performed by the processor.

3.3 Operating System

A decision was taken to use Tripos [Richards79, Wilson85], a light-weight multi-tasking operating system written in BCPL, originally developed at the University of Cambridge. There were four main reasons for chosing Tripos as the vehicle for video experimentation. Firstly, it is small system, based on message passing, designed specifically for the development of real-time applications. All tasks run in the same address space, and there is no form of memory protection. As a result, context switches are extremely fast, and this combined with a pre-emptive scheduling strategy based on tasks having unique priorities, makes the system ideally suited for time critical applications such as video. Secondly, Tripos has been used successfully as the basis for many of the distributed applications in the Cambridge Model Distributed System, and hence it has a proven track record in a real-time network environment. Thirdly, being a locally developed system, there is a large amount of knowledge as to how to use Tripos, and how to develop applications for it. Finally, and probably most importantly considering the nature of our research, the sources to the entire system are freely available.

3.4 Cameras

Although video cameras can be considered as standard, off-the-shelf components, it was surprisingly difficult to find ones which were appropriate for the needs of our experimentation. Any camera associated with a computer workstation had to be small and easily mounted—ideally directly above the screen—and had to be cheap enough to be treated like any other peripheral.

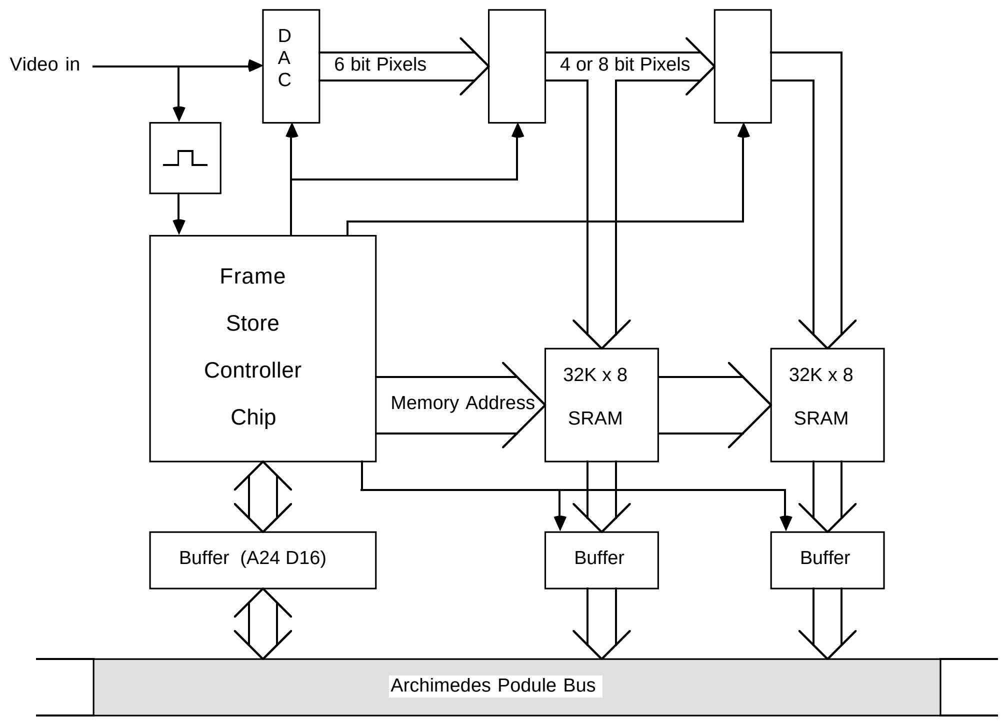
Other than the extremely expensive studio cameras, there were only two categories of product on the market: "home" video and CCTV surveillance. Cameras intended for the making of home movies were ideal for our work, with all the necessary functionality, and often other useful features such as automatic focus. Unfortunately, home video cameras usually come with an integrated VCR, which has the effect of making them artificially expensive. CCTV cameras are much cheaper, but of lower quality, and have the tendency of being inflexible. Being designed mainly for surveillance work, CCTV cameras are intended to be fixed permanently to a wall or ceiling. Consequently, there is a tendency for the focus to be either fixed by the manufacturer or difficult to change, and for the exposure to be controlled with an automatic iris or AGC. As with other parts of the system, it was necessary to compromise in order to find something suitable for our needs at a reasonable cost; in the end, we settled on two small CCD surveillance cameras, one made by Sony and the other by Philips.

3.5 Frame Capture Hardware

The hardware was designed to be capable of capturing digital images in real time. As mentioned earlier, the Archimedes has no DMA facilities; hence, any frame capture hardware had to be capable of buffering the image being digitized, so that it could be read out in an efficient manner by the processor at the end of the frame. In order to cut down on the amount of logic needed to control the frame buffer, a custom Frame Store Controller chip was developed [Milway88]; this chip generates all the signals necessary to address the frame buffer memory, and provides hardware support for the sub-sampling of images as they are digitized.

The resulting Frame Capture board for the Archimedes is small—the chip count is only 15, including the Frame Store Controller and two SRAM chips—and is capable of digitizing up to 256×256×8 monochrome images. The analogue composite video signal is fed into the board through a BNC connector (or equivalent), and digitizing circuitry on the board synchronizes to the frame rate of the camera. The board will capture either or both of the odd/even fields of a frame, and is capable of running in a mode where consecutive fields are captured repeatedly, for use when digitizing a series of frames. The only part of the board which is at all Archimedes dependent is the bus interface.

Video Frame Store Controller



4. First Applications

Having described the preparatory work which was necessary in order to create our video development environment, it is now possible to look at the first experimental applications which were built using this system. In choosing the initial applications—Photo Booth, Movie Capture and Videophone—we wanted to find some which gave a flavour of how video might be integrated into the workstations of the future, while at the same time minimizing the complexity so as to be able to produce the applications in a reasonable time scale.

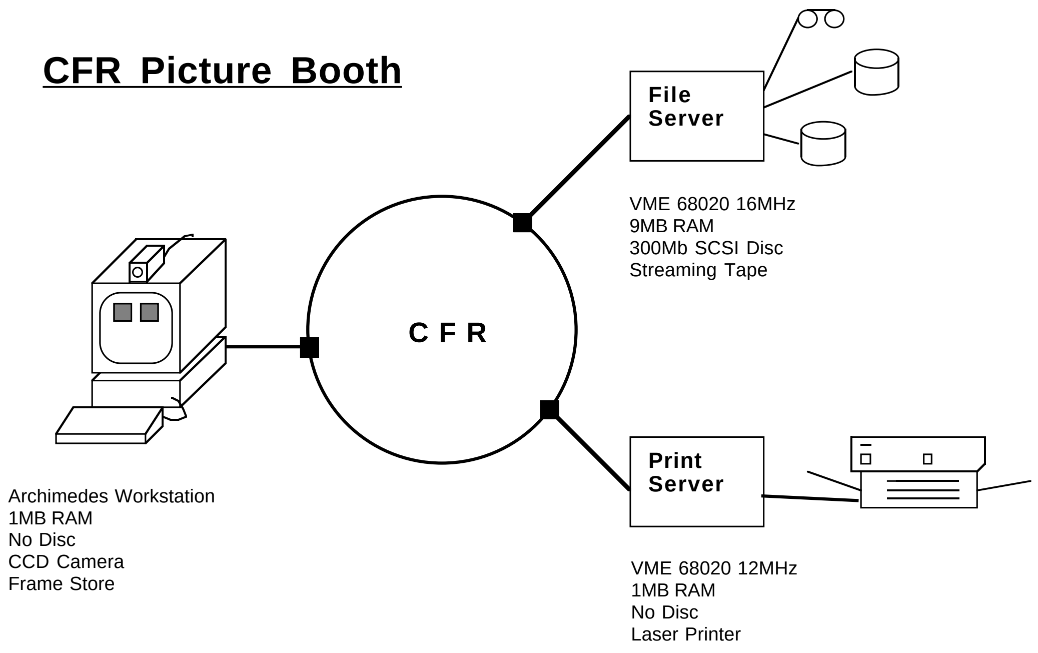
4.1 Photo Booth

One obvious advantage of attaching a video camera to a workstation is that it can be used to take still pictures, which are then stored as part of the computer's filing system—maybe as part of some database. An obvious advantage of capturing images using a camera rather than a scanner is that there is no restriction on the size, shape or condition of the target object; in particular, a camera can be used to capture three dimensional objects, for example, faces.

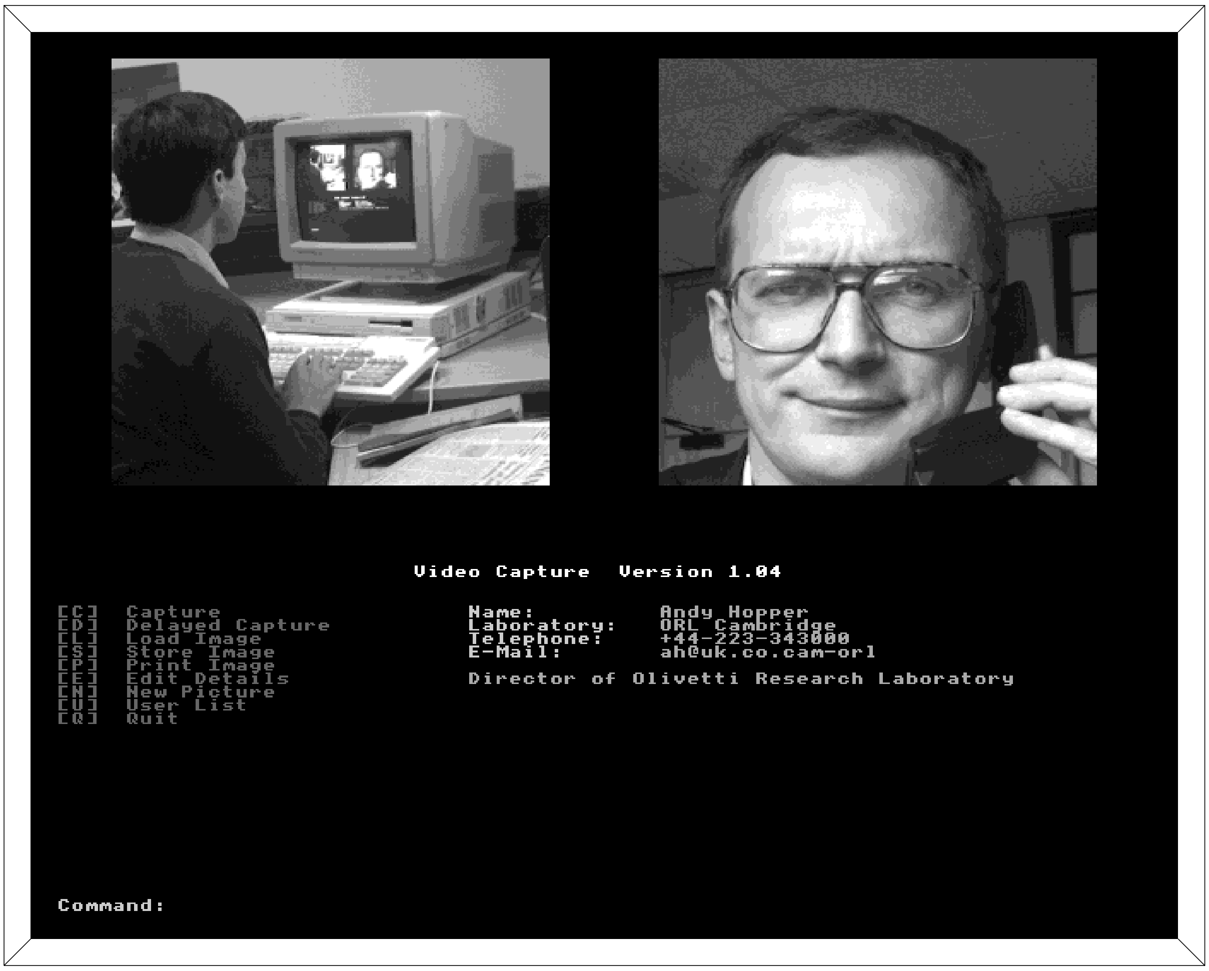
The first application was intended to show how this might work in practice, and takes the form of an electronic Photo Booth, analogous to those used to take passport photographs. There was, in fact, a requirement for such an application with visitors to our own laboratory: being capable of capturing their pictures made it possible to store the images in a database, along with names, addresses, telephone numbers and so on; a kind of *aide-mémoir* which made it possible to attach names to faces.

The Photo Booth hardware setup is an Archimedes with a camera, Frame Capture board and CFR interface. The CFR is used for access to network services, particularly the File

Server and Print Server. Images are captured at 25 fps, and displayed on the left hand side of the screen; this allows the user being photographed to position himself correctly, and sort out problems such as focus and lighting. When ready, the user gives the command to take a photograph; a 5 second countdown is started, at the end of which a single frame is captured, and displayed on the right hand side of the screen. The user can then use the keyboard to fill in the information fields associated with the picture; when this is complete, the information is stored together with the image on the CFR File Server. Stored images can be retrieved from the File Server and displayed on the screen, or converted to PostScript and sent to the Print Server for printing on a Laser Writer.



There are several aspects of the Photo Booth which are worth discussing here. Because the real time video is only being manipulated locally, there is much less of a bandwidth limitation. Consequently, 256×256 images are used, allowing 2 pictures to be viewed side by side on the screen using the highest resolution 4 bit mode. The only time the images are transmitted over the CFR is to or from the remote File Server: a task which does not have to be accomplished in real time. Secondly, the Frame Store Controller chip allows the capture of consecutive frames at different resolutions. This means that, even though the Archimedes itself is only capable of displaying images which are 4 bits deep, the still images could be captured at the limit of the A/D converter (in this case 6 bits). Finally, because the library of captured pictures is held on the network File Server, it can be shared by any number of machines on the CFR. This means that there can be several people capturing pictures simultaneously, and that any computer attached to the CFR, whether or not it has a camera and frame store, is capable of accessing the library of images.



```
                         Video Capture  Version 1.04

[C]  Capture                     Name:          Andy Hopper
[D]  Delayed Capture             Laboratory:    ORL Cambridge
[L]  Load Image                  Telephone:     +44-223-343000
[S]  Store Image                 E-Mail:        ah@uk.co.cam-orl
[P]  Print Image
[E]  Edit Details                Director of Olivetti Research Laboratory
[N]  New Picture
[U]  User List
[Q]  Quit

Command:
```

Screen Dump taken from the Photo Booth

4.2 Movie Capture

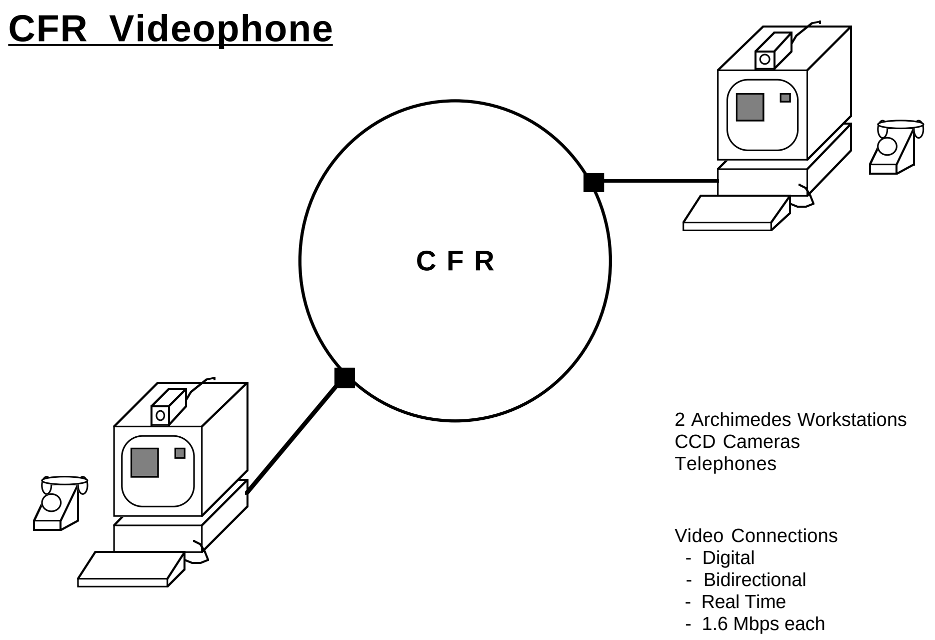
The second experimental application was intended to give us an idea of the issues involved in capturing video sequences, and in particular, the real time constraints which this placed on the software. We also wanted to gain experience of what it would be like sitting in front of a camera, presenting a monologue—something which would be an integral part of, for example, a video mail system.

The main problem with capturing video sequences is the sheer amount of data generated. Even using what we felt was the lowest reasonable resolution, 128×128×4, each frame is 8K bytes, and at 25 fps, data is generated at the rate of 200K bytes per second. The CFR File Server, being designed as a general purpose file repository, was only capable of supporting around 120K bytes per second, and then only if being used by a single client. Without designing a totally new File Server, it was clear that we would not be able to store uncompressed video data in real time, so we adopted the compromise of recording directly into the RAM of the Archimedes, and then spooling the data out to the File Server when the recording was complete. Alternatively, we could have performed simple real-time compression of the images being captured (for example, only working with frame differences), or even reduced the size of the image still further. The largest configuration of Archimedes has enough memory to record around 15 seconds of uninterrupted video, and we judged this to be sufficient for our purposes.

The same hardware configuration as the Photo Booth was used to capture the movies, and they were stored on the same File Server. Facilities were added to view the movies in real time, and speeded up either forwards or backwards. As with the Photo Booth, two images were displayed on the screen simultaneously: the left hand image being the live image from the camera, and the right hand image being the one captured previously or retrieved from the File Server. This ability to store and retrieve video sequences from a shared File Server is the basis for many potential applications, including electronic video mail.

4.3 Videophone

The third application was a simple Videophone, which allowed for the bidirectional transfer of real time video images. Only one-to-one video connections were contemplated; conferencing, in other words one-to-many or many-to-many connections, has its own special set of problems, many of which are totally unrelated to the fact that video is involved. Given that the ultimate aim of our work was to investigate the integration of video into the computer environment, we deliberately avoided any aspects which might potentially have diverted us from this path. Probably the most important reason for keeping to a simple Videophone was that it enabled us to reduce the amount of implementation effort needed, and thus allowed the application to be written in a matter of weeks rather than months.



The Videophone requires two Archimedes computers, each with a Frame Capture board and a CFR connection. Each machine captures images in real time, and transmits them over the network to the other. So as to give each user the ability to position himself in front of his own camera, a smaller version of the transmitted image is also displayed on the local screen. In combination with a simple analogue telephone, the effect is bidirectional Videophone.

As mentioned earlier, the amount of data generated by a video application is very large, and hence it was necessary to reduce the size of the image to be transmitted to 128×128×4. At an update rate of 25 fps, this requires 1.6 Mbps for each of the two video channels; hence, each of the CFR stations had to maintain an effective bidirectional throughput of 3.2 Mbps—close to the theoretical limit of the Archimedes CFR interfaces.

As an experiment, a very simple compression algorithm was applied to the video data. The 32 bytes of the CFR packet were split into 4 bytes for control information, and 28 bytes for data. The image to be transmitted was then split up into "tiles" of 7×8 4 bit pixels, and each tile was encoded in a single packet. Before being transmitted, each tile was compared with its previous value, and only transmitted if any of the pixels within the tile had changed. In addition to the frame differences the whole image was sent 4 times per second, in order to cope with packets lost due to transmission errors or receiver contention.

5. Experiences and Feedback

Given the initial aims of our research, the most important results of the work are the experiences gained whilst the work was in progress, and the feedback obtained from the people who have viewed the work. The simplicity of some of the observations surprised us. Many of the problems we encountered—subject lighting, for instance—were nothing to do with the fact that we were working with digital video, but were of a much more general nature.

5.1 Cameras

While fixed focus cameras are fine for certain applications, such as a video telephone, they are very restrictive in others; for example, they cannot be used to take close up shots of documents. One solution is to fit the camera with a wide angle lens, which has the effect of increasing the effective depth of field, but at the expense of adding distortion and making the objects in the picture appear much further away. Variable focus cameras with manual operation avoid this particular problem, but present another when placed by a workstation and pointed at its user. In order to focus on that user, the lens must be adjusted, and this usually involves the user in reaching forward. Unfortunately, if the camera is focussed on the user in this position, the image is out of focus when the user leans back to the normal position. The obvious solution is to use lenses with automatic focus, but they have the disadvantage of being expensive.

CCD cameras are cheap, light, and require very little power; hence, they are well suited for our kind of application. One problem with them, though, is that their response to light of various frequences is different to that of the eye; in particular, they are sensitive to light in the infra-red part of the spectrum. In situations such as direct sunlight or high power spotlights, certain objects (paticularly dark cloth, which reflects infra-red radiation) often appear much lighter than they should.

5.2 Lighting

As with any other photographic technique, lighting is the key to a good picture. Unfortunately, in the context of computer video, it is not always possible to solve the lighting problems satisfactorily. For example, it is not feasible to fit every office or laboratory with arc-lights to ensure even illumination of the subjects; nor is it always possible to move equipment relative to the windows in a room. It is also the case that cameras with automatic exposure control tend to be fooled by certain situations. Until we changed to a manual aperture setting, people wearing white shirts always appeared much more sun-tanned than their dark shirted counterparts. Automatic cameras are also confused by images which are in any way reflective, for example glass and shiny metal. Using cameras with manual exposure control solves many of the problems, but must continually be adjusted in rooms which, depending on the weather and time of day, may or may not be in sunshine. In future, it may be necessary for the user to control camera exposure remotely from his workstation.

5.3 To reflect, or not to reflect?

One of the problems we have found in our work with video is that, when a user is faced with an image of himself, because of his experience with mirrors, he expects the image to be a reflection. When presented with the image which the camera sees, horizontal movements are reversed, and this makes it surprisingly difficult for the user to position himself in the centre of the picture; complicated operations (such as tying a tie) are well nigh impossible. The obvious answer is to reflect the image as it is captured, and since this only means reading the CCD array in reverse order, many cameras offer this as an option. Although this keeps the user happy with respect to the image he sees, a reflected image is useless for anyone else, since they are expecting to see a true image, and are confused by the reflection.

From a functionality point of view, it is fairly obvious what is needed: images from a camera should be reflected when displayed locally, but not when displayed remotely. Unfortunately, this means that it is impossible to take advantage of the reflection capability in the camera, since the image is required in both normal and reflected forms. Unless specialised hardware is available for the purpose, the reflection must be performed in software, as the image is being written into the display buffer. The amount of processing required for this, conceptually simple, operation should not be underestimated—for example, in the Archimedes case with 4 bit pixels, not only is it necessary to re-order the bytes in each scan line, but also to swap the nibbles in each byte. The effect is that, even with a processor as fast as an ARM, it is very difficult to perform such an operation in the time period between two frames, in Europe 40 mS.

5.4 Video quality

Throughout our work, we have been using video images which are significantly smaller and lower in quality than normal TV pictures. We have always used square (either 128×128 or 256×256) 4 bit monochrome images. The reactions to quality of the video images have been mixed but, on the whole, favourable. Low resolution images, although rather poor, are acceptable so long as the full frame rate is maintained. The eye compensates for the lack of quality, so long as there is movement in the image. As soon as the movement stops, the perceived quality drops: 128×128 moving images appear of similar quality to 256×256 stills. The higher resolution images appear significantly better, and assuming the lighting is set up in such a way as to maximize contrast, it is barely noticeable that there are only 16 grey levels in the displayed images. The quality of the perceived pictures can be improved still further by manipulating the graphics palette, for example by reducing the number of levels displayed when viewing an image containing text, or assigning the display levels in a non-linear manner to correct for lighting deficiencies.

In an attempt to investigate how important the quality of the displayed video was, a Video Overlay board was built. The board comprised a frame buffer, capable of holding the video images at their full resolution, a D/A converter, and some video switch circuitry. Rather than writing the picture directly into the frame buffer of the Archimedes, it was written into the buffer memory of the Video Overlay board. The analogue video output was taken from the Archimedes, and mixed with the output of the overlay frame buffer; from the user’s point of view, the only difference is that the video is always displayed at its full resolution, which is independent of the Video Controller within the Archimedes. Surprisingly, although the 6 bit image was better than the 4 bit version, it was not considered an "essential" improvement by those who saw it. It turns out that the video overlay technique is attractive for other reasons—it divorces the video buffer from the host frame buffer, thus increasing hardware independence and reducing bus loading—and so we have adopted it for our future work.

5.5 Reactions to the applications

It seems that George Orwell, among others, has instilled in us an innate hatred of cameras. Unfortunately, it is not possible to avoid the psychological aspects of video work, and to many people, the presence of a video camera in a room is rather too close to "big brother" for comfort. We have had negative reactions to our work, albeit rather few. Anyone proposing to exploit video technology will have to take this attitude into account, if it is not to be rejected on the grounds of being socially unacceptable. In order to reduce the threatening impact of a video camera, the user must be in full control of where the camera is pointing, and whether it is even switched on.

Video telephones were treated with scepticism by almost everybody, irrespective of their background. Everyone came up with reasons for disliking the idea—common questions were "what happens when I am in the bath?" and "how can I tell a salesman to *go away* to his face?"—and virtually nobody could think of anything positive to say. Earlier pilot studies such as the Picturephone system [Bell71] reported similar reactions.

The Videophone application was naïve in nature, but was sufficiently real to show that the experimental system we had chosen had the power to perform the tasks required of it; in particular, the CFR was capable of carrying the video traffic in real time. The rather poor quality of image was adequate for the application since the human eye is fooled by motion, giving an improved perceived image. Ironically, because the images were of relatively low resolution, the problem of "eye contact" between the users of the system never really arose; it was not possible, with any accuracy, to see which way the eyes were pointing.

On a more positive note, there was a lot of interest in the Photo Booth application. Not because it was especially useful in itself, but because it demonstrated two important features: firstly, it is possible to store images in a computer’s filing system; secondly, it was possible to treat a video camera as a general purpose input peripheral. As an example of the second point, it is possible to use a camera to digitize items which cannot go through a scanner, either because they are three dimensional, or because they are particularly brittle, such as old manuscripts or paintings—interest has already been expressed by a museum to use our system to photograph objects as diverse as a Roman coin and a London bus. We believe that, in the future, a camera will be as common a peripheral on a workstation or personal computer, as a mouse is today.

6. Conclusion

This paper represents early experimental work in the area of computer video. Our aim was to put together sufficient hardware and software to demonstrate the feasibility of video applications, and to gain experience in the problems involved whilst doing so. We have deliberately attempted to use the applications within our own laboratory, and to expose as many different types of people as possible to the work we have been doing. In this way, we believe that we have gained useful insights into the use of video in computer applications, and the feedback has been sufficient to encourage us to continue in this area.

As a result of the early video work described here, we now have a rich environment for starting further multi-media work. Integration of FAX and Scanners, a Video Editor and Electronic Video Mail are just three applications which will follow on from these beginnings. Work has already started on the development of *Pandora*, a "magic" video box which integrates into a single unit much of the earlier hardware and software effort. From now on, except for specific pieces of experimentation, all video hardware is being built for maxium machine independence, so as to open up the possibility of video applications to as many users as possible. Work has also started to add sufficient extensions to the X Windows system [Scheifler88, Levy88] to enable the Pandora facilities to be controlled from remote X clients. Without the early experimental video work presented here, none of this would have been possible.

7. Acknowledgements

The authors would like to thank the other members of the Olivetti Research Laboratory for their help in the early video work. In particular, Oliver Mason has been heavily involved in work the analogue parts of the video circuitry, and Tony King was responsible for the Videophone application.

8. References

[Acorn87] Acorn Computers Ltd.
Archimedes User Guide

[Bell71] Bell Systems Research Laboratories
The Picturephone System
Bell Systems Technical Journal, February 1971

[Coudreuse88] J.-P. Coudreuse
ATM: State of Definition and Major Issues
Proceedings Second International Workshop on Packet Video, 1988

[Davenport88] G. Davenport
Software Considerations for Multi-Media Video Projects
Proceedings X11 Video Extension Technical Meeting, 1988

[Hopper88] A. Hopper and R. M. Needham
The Cambridge Fast Ring Networking System
IEEE Transactions on Computers 37 No. 10, 1988

[Milway88] D. R. Milway
The Video Capture Controller Chip
Olivetti Research Laboratory Cambridge Technical Report, 1988

[Needham82] R. M. Needham and A. J. Herbert
The Cambridge Distributed Computing System
Addison Wesley, 1982

[Lampe88] D. R. Lampe
Athena Muse: Hypermedia in Action
The MIT Report, February 1988

[Levy88] M. Levy
X Video 'C' Programming Interface, X Video Extensions
Parallax Graphics Technical Report, 1988

[Parallax87] Parallax Graphics
The Parallax 1280 series Videographic Processor
Parallax Graphics Technical Report, 1987

[Richards79] M. Richards, A. R. Aylward, P. Bond, R. D. Evans and B. J. Knight
TRIPOS: A Portable Operating System for Mini-Computers
Software Practice and Experience 9 No. 7, 1979

[Scheifler88] R. W. Scheifler
The X Window System Protocol
MIT Laboratory for Computer Science Technical Report, 1988

[Want88] R. Want
Reliable Management of Voice in a Distributed System
University of Cambridge Computer Laboratory Technical Report 141, 1988

[Wilson85] I. D. Wilson
Operating System Design for Large Personal Workstations
University of Cambridge Computer Laboratory Technical Report 83, 1985